Techniques for Testing Experimental Network Protocols

Brian L. Tierney (bltierney@lbl.gov)
Jason Lee

Distributed Systems Department
Lawrence Berkeley National Laboratory

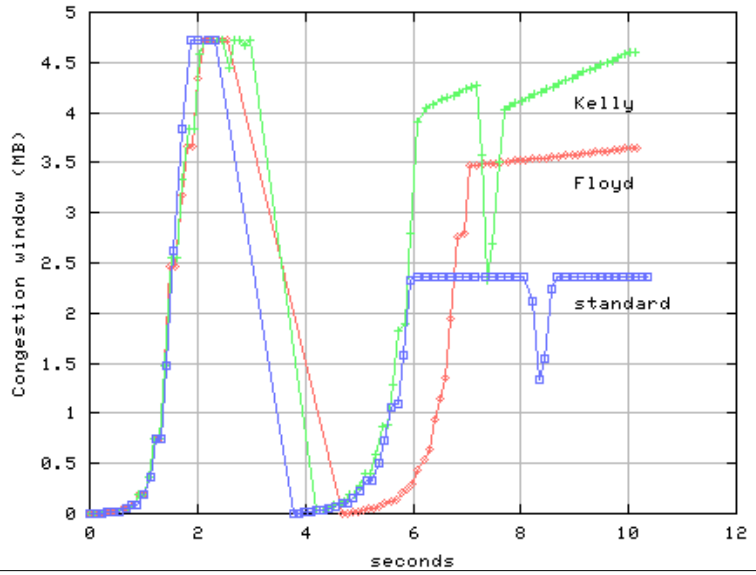
The Problem

- Very hard to draw real conclusions from the results in most PFLDnet submissions
- Lack of problem description:
  - Exactly what are your tests trying to prove?
- Results often not generalize-able
- Statistical Analysis often very weak

U.S. Department of Energy
Office of Science

Typical Results

BERKELEY LAB



3

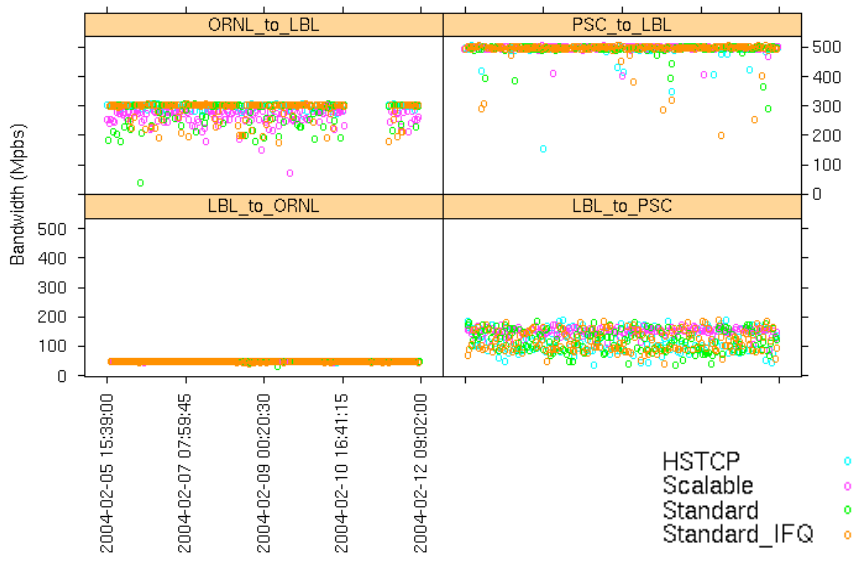
U.S. Department of Energy
Office of Science

What do we learn from this plot?

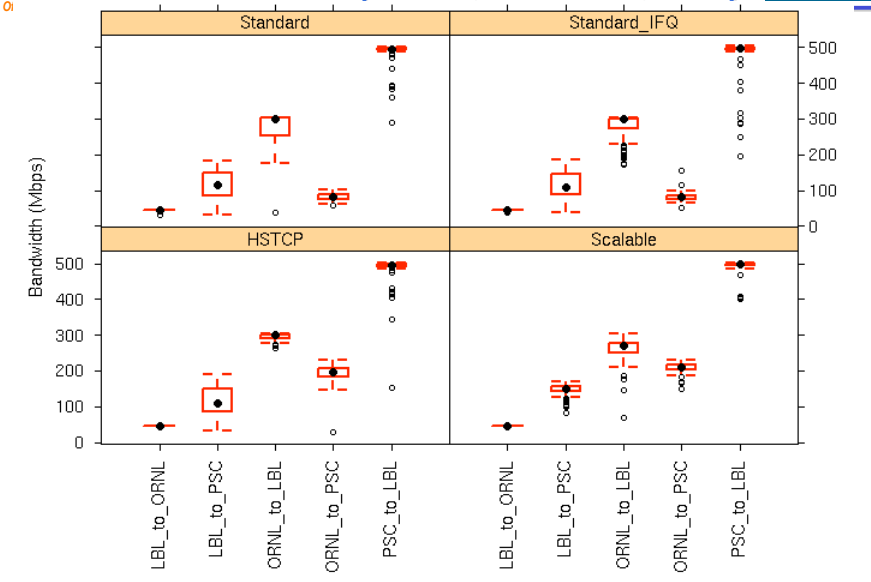
BERKELEY LAB

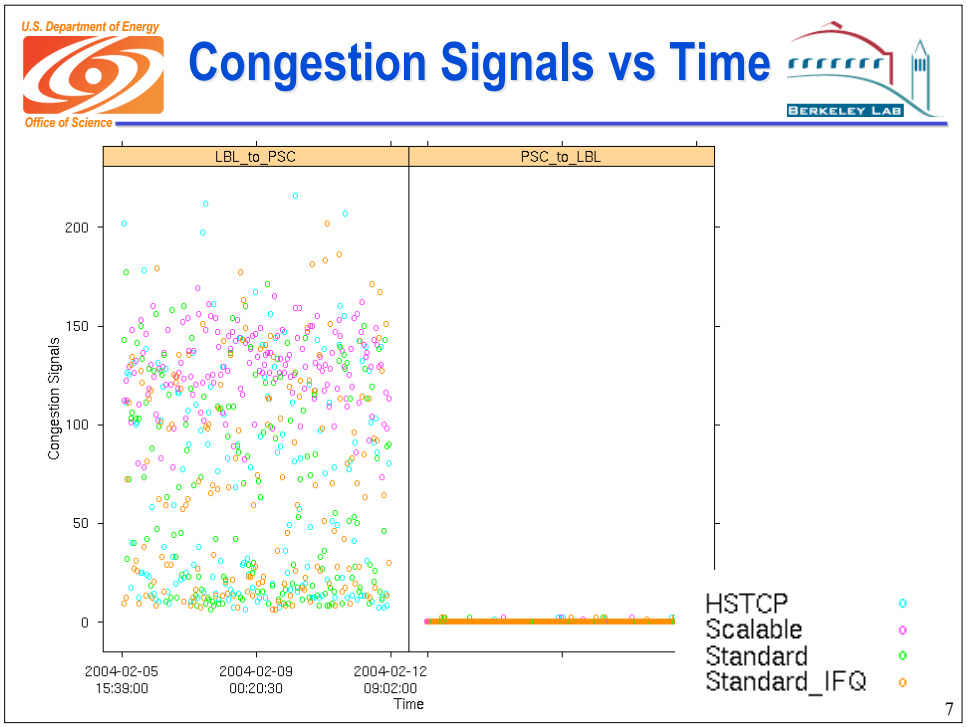
- Very little!!
  - CWND vs time using different algorithms on some unknown path on a single random test?
- What are the path characteristics?
  - RTT, available bandwidth, number of hops, RED, etc?
  - real network or “testbed”?
  - Is the path symmetric?
- What testing methodology was used?
  - How many tests?
  - For how long?
  - Day vs Night vs weekday vs weekend?

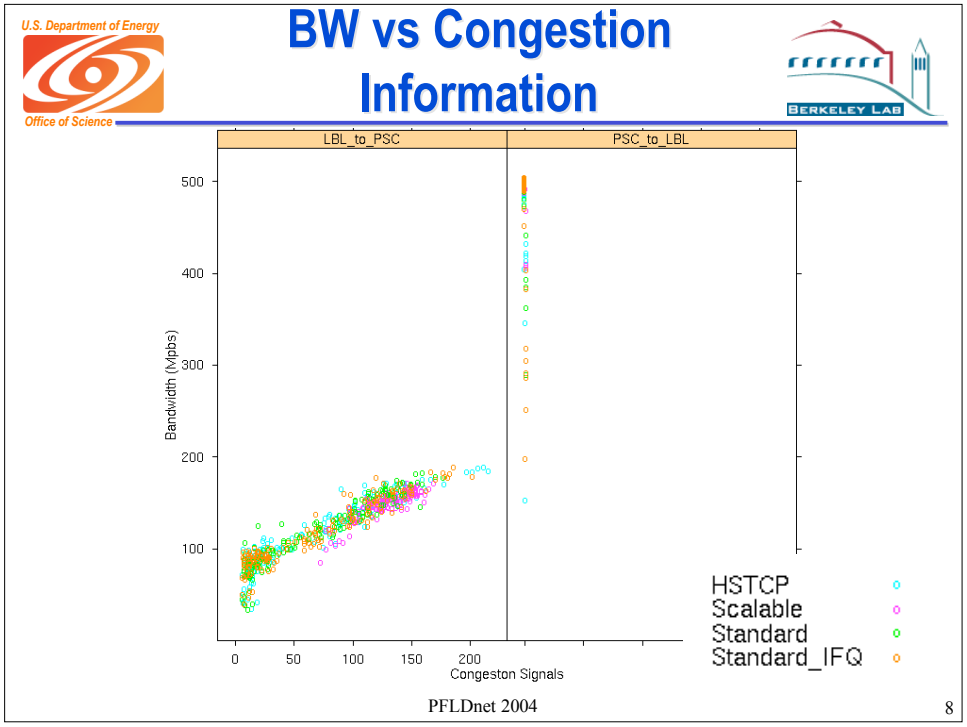
Better: 1 week of results, multiple paths



Even Better: add some statistics (mean, Std Dev, etc.)




U.S. Department of Energy
Office of Science
Congestion Signals vs Time
BERKELEY LAB
LBL_to_PSC
PSC_to_LBL
Congestion Signals
200
150
100
50
0
2004-02-05
15:39:00
2004-02-09
00:20:30
2004-02-12
09:02:00
Time
HSTCP
Scalable
Standard
Standard_IFQ
7


U.S. Department of Energy
Office of Science
BW vs Congestion Information
BERKELEY LAB
LBL_to_PSC
PSC_to_LBL
Bandwidth (Mpbs)
500
400
300
200
100
0
50
100
150
200
Congeston Signals
HSTCP
Scalable
Standard
Standard_IFQ
PFLDnet 2004
8

Why doing is "right" is Hard

- Hard to get accounts on enough hosts
  - Projects like Planetlab helping with this issue
- Hard to get custom kernels installed
- Hard to distinguish host effects from network effects
  - Might be other processes running that effect results
- Hard to schedule tests to not overlap

Possible Solutions

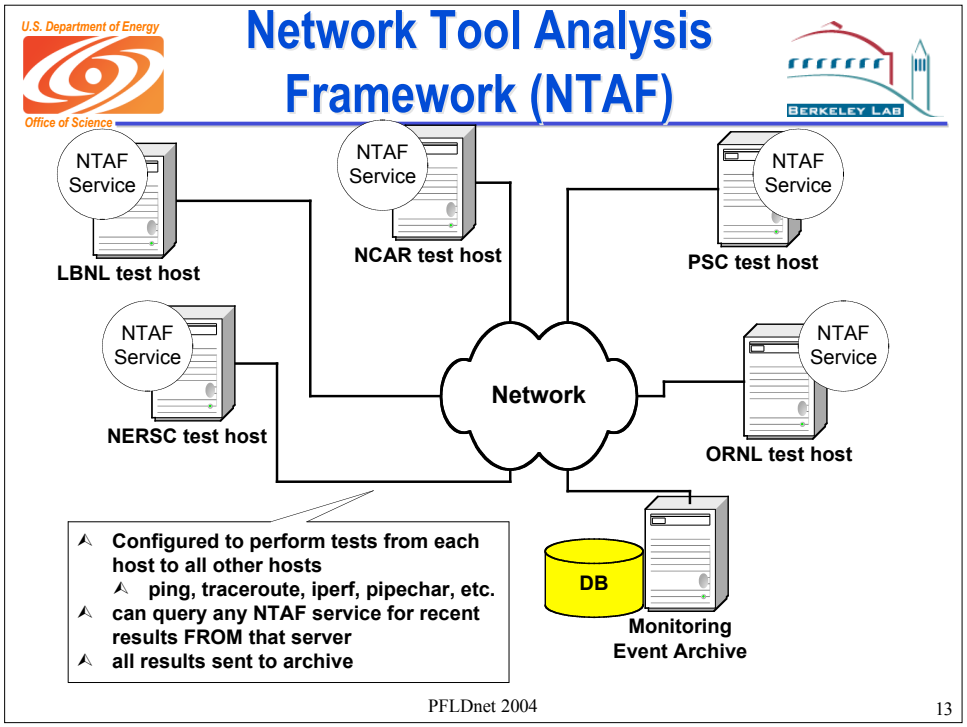
- Simulation (e.g.: ns-2)
- Emulation e.g.(Emulab)
- Run lots of tests by hand
- Use lots of "cron" jobs

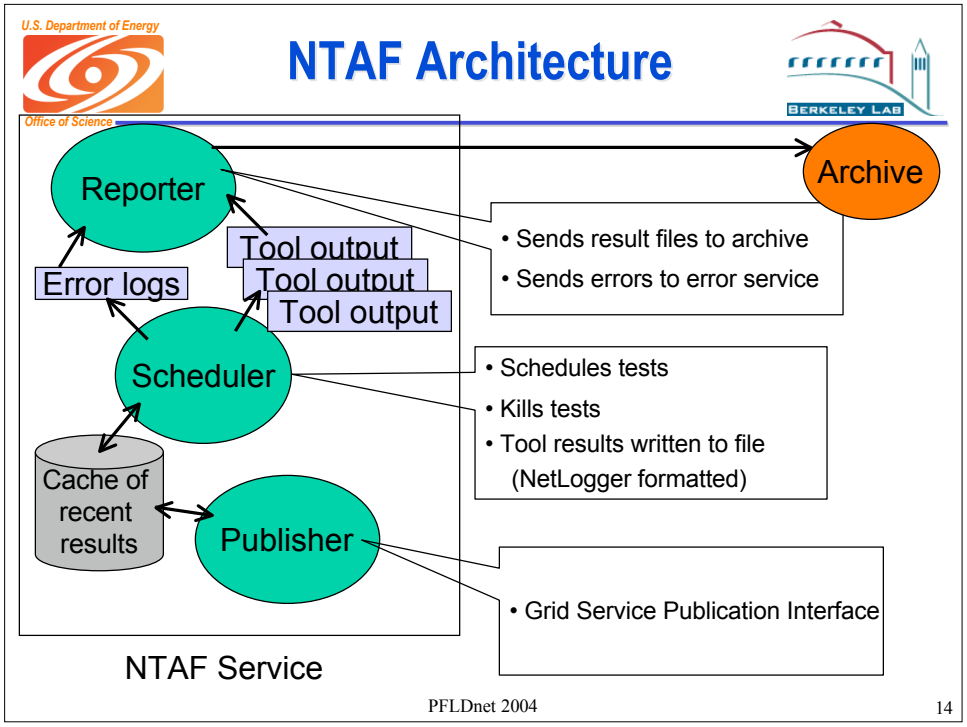
Another Solution

- A testing framework that facilitates
  - Collection over multiple paths
  - Regular testing for extended periods of time
  - Inclusion of host monitoring data
  - Collection of results to a central location for analysis
  - Insertion of results into a relational database

Network Tool Analysis Framework (NTAF)

- Configure and launch network tools on a predefined schedule
  - measure bandwidth/latency (iperf, pchar, pipechar)
  - augment tools to report Web100 data
- Collect and transform tool results into a common format
- Uses NetLogger to format and send data to archive
  - Wrap existing tool with a python script to parse results and convert into a NetLogger message
    - ping, iperf, pipechar, pathrate, pathload, netest, traceroute, etc.
  - Easy to add new tools: only need to write python wrapper


U.S. Department of Energy
Office of Science
Network Tool Analysis Framework (NTAF)
BERKELEY LAB
NTAF Service
LBNL test host
NTAF Service
NCAR test host
NTAF Service
PSC test host
NTAF Service
NERSC test host
Network
NTAF Service
ORNL test host
Configured to perform tests from each host to all other hosts
ping, traceroute, iperf, pipechar, etc.
can query any NTAF service for recent results FROM that server
all results sent to archive
DB
Monitoring Event Archive
PFLDnet 2004
13


U.S. Department of Energy
Office of Science
NTAF Architecture
BERKELEY LAB
Reporter
Archive
Tool output
Tool output
Tool output
Error logs
• Sends result files to archive
• Sends errors to error service
Scheduler
• Schedules tests
• Kills tests
• Tool results written to file (NetLogger formatted)
Cache of recent results
Publisher
• Grid Service Publication Interface
NTAF Service
PFLDnet 2004
14

NTAF Design Issues

- **Fault tolerance: Many test tools are experimental and may crash or hang.**
  - NTAF is designed to insure that a tool does block the NTAF server.
  - Each test is run in its own thread, with a time-out to ensure the test eventually ends.
  - All socket I/O is non-blocking to ensure that nothing ever blocks waiting for a message that may never arrive.
- **Automatic restart: Certain tools (e.g.: iperf) require a remote server.**
  - These servers will sometimes crash or hang, and must be monitored
  - NTAF can be configured to run tests to localhost servers, and call the restart script if the test fails.

NTAF Design Issues

- **Output formats: Every tool has a different output format, which can be difficult to parse.**
  - We use simple python wrappers around each tool to generate NetLogger events.
  - NTAF components only needs to understand NetLogger formatted data.
- **Test scheduling: A flexible scheduling mechanism is required.**
  - Some tests are very intrusive, and should not be run often.
  - Other tests must be run in isolation, or at least with no other similar tests (e.g.: pipechar)
  - To avoid possible test synchronization effects, we add a randomization factor to the scheduling.
    - E.g.: run a test every 90 minutes plus or minus a random time between zero and five minutes.

Relational Network Monitoring Data Archive

- A relational database that supports SQL is an excellent tool for data mining of network monitoring data.
- SQL provides a general and powerful language for extracting data.
- For example, with SQL we can easily do queries such as:
  - find the average available bandwidth for the past 100 tests
  - return all data for tests runs where the throughput was less than 50% of the average throughput
  - return all data for tests runs where the delay was more than double the average delay
  - return all data for time period where CPU load < 50%

NTAF Limitations

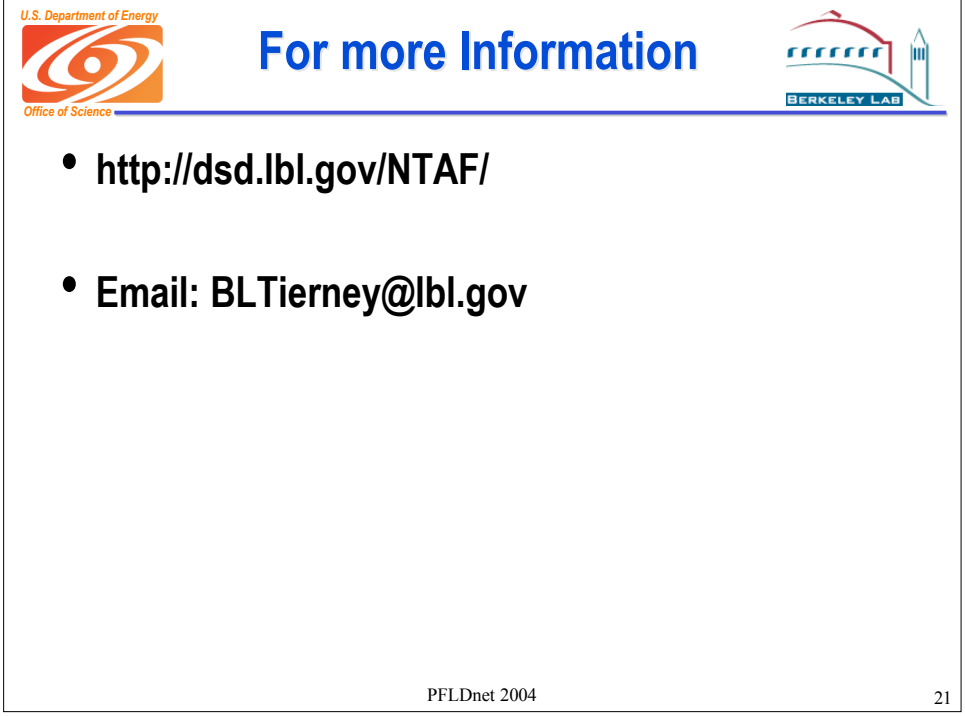
- No global scheduling
- Servers are all independent
  - Adds reliability, but reduces functionality
  - E.g.: can't send message to server to change some receiver setting
- Limited support for co-scheduling
  - E.g.: run multiple tests at once to test for fairness
- More host monitoring is needed
  - Currently just CPU

Conclusions

- Intent of this talk is *not* to make everyone start "fixing" their PFLDnet presentations now :-)
- Intent is to get the community to think harder about these issues, and come up with some guidelines
- Use standard scientific methods:
  - State hypothesis, collect evidence, draw conclusion
  - Often need better "controls"

Possible List of Guidelines (for discussion)

- Try to provide some simulation results
- Try to provide results for a variety of real path
  - Fast, slow, with/ without congestion, short/long RTT, etc.
  - Run at least 50? tests over a 1 week period?
  - Always test bi-directional
- Report statistics (e.g.: mean and standard deviation)
- Try to monitor the hosts and factor out host-induced outliers
- Always include both bandwidth and congestion data


U.S. Department of Energy
Office of Science
For more Information
BERKELEY LAB
• http://dsd.lbl.gov/NTAF/
• Email: BLTierney@lbl.gov
PFLDnet 2004
21